Evaluating Weakly Supervised Object Localization Methods Right – Supplementary Material –

We include additional materials in this document. Each section matches with the main paper sections: §A to main paper §3, §B to main paper §4, and §C to main paper §5.

A. Problem Formulation of WSOL

A.1. CAM as patch-wise posterior approximation

In the main paper §3.1, we have described class activation mapping (CAM) [13] as a patch-wise posterior approximator trained with image-level labels. We describe in detail why this is so.

Equivalent re-formulation of CAM. Originally, CAM is a technique applied on a convolutional neural network classifier $h : \mathbb{R}^{3\times H\times W} \to \mathbb{R}^C$, where C is the number of classes, of the following form:

$$h_c(X) = \sum_d W_{cd} \left(\frac{1}{HW} \sum_{ij} g_{dij}(\mathbf{X}) \right) \tag{1}$$

where c, d are the channel-dimension indices and i, j are spatial-dimension indices. In other words, h is a fully convolutional neural network, followed by a global average pooling (GAP) and a linear (fully-connected) layer into a C-dimensional vector. We may swap the GAP and linear layers without changing the representation:

$$h_c(X) = \frac{1}{HW} \sum_{ij} \left(\sum_d W_{cd} g_{dij}(\mathbf{X}) \right) \tag{2}$$

$$=: \frac{1}{HW} \sum_{ij} f_{cij}(\mathbf{X}) \tag{3}$$

where f is now a fully-convolutional network. Each pixel (i, j) in the feature map, $(f_{1ij}(\mathbf{X}), \cdots, f_{Cij}(\mathbf{X}))$, corresponds to the classification result of the corresponding field of view in the input $\mathbf{X}$, written as X_{ij}. Thus, we equivalently write

$$h_c(X) = \frac{1}{HW} \sum_{ij} f_c(X_{ij}) \tag{4}$$

where f is now re-defined as a image patch classifier with 1-dimensional feature output (not fully convolutional).

CAM as patch-wise posterior approximator. h is now the average-pooled value of the patch-wise classification scores $f_c(X_{ij})$. CAM trains f by maximizing the image-wide posterior of the ground truth class, where the posterior is defined as the softmax over $f(X_{ij})$:

$$\log p(Y|\mathbf{X}) := \log \text{softmax}^Y \left(\frac{1}{HW} \sum_{ij} f(X_{ij}) \right). \tag{5}$$

In other words, CAM trains the network for patch-wise scores $f_c(X_{ij})$ to estimate the image-wide posterior $p(Y|\mathbf{X})$.

At inference time, CAM estimates the pixel-wise posterior $p(Y|X_{ij})$ approximately by performing $p(Y|X_{ij}) \approx f_Y(X_{ij}) / \max_{\alpha\beta} f_Y(X_{\alpha\beta})$ (modulo some calibration to make sure $p(Y|X_{ij}) \in [0, 1]$).

A.2. Proof for the ill-posedness lemma

We first define an evaluation metric for our score map for an easier argumentation.

Definition A.1. *For a scoring rule s and a threshold τ, we define the **pixel-wise localization accuracy** $\texttt{PxAcc}(s, \tau)$ as the probability of correctly predicting the pixel-wise labels:*

$$\begin{aligned}\texttt{PxAcc}(s,\tau) = P_{X,T}(s(X) \geq \tau \mid T = 1) \cdot P_{X,T}(T = 1) \\ + P_{X,T}(s(X) < \tau \mid T = 0) \cdot P_{X,T}(T = 0)\end{aligned}$$

We prove the following lemma.

Lemma A.2. *Assume that the true posterior $p(Y|M)$ with a continuous pdf is used as the scoring rule $s(M) = p(Y|M)$. Then, there exists a scalar $\tau \in \mathbb{R}$ such that $\texttt{PxAcc}(p(Y|M), \tau) = 1$ if and only if the foreground-background posterior ratio $\frac{p(Y=1|M^{fg})}{p(Y=1|M^{bf})} \geq 1$ almost surely, conditionally on the event $\{T(M^{fg}) = 1$ and $T(M^{bf}) = 0\}$.*

Proof. We write $E := \{T(M^{\text{fg}}) = 1 \text{ and } T(M^{\text{bf}}) = 0\}$.
(**Proof for "if"**) Assume $\alpha \geq 1$ almost surely, given E. Let

$$\tau := \min_{G:P(G\Delta\{T(m)=0\})=0} \max_{m\in G} p(Y = 1|M = m) \tag{6}$$

Figure 1: **Ducks.** Random duck images on Flickr. They contain more lake than feet pixels: $p(\text{water}|\text{duck}) \gg p(\text{feet}|\text{duck})$.

where Δ is the set XOR operation: $A \Delta B := (A \cup B) \setminus (A \cap B)$. Then, for almost all $M^{\text{fg}}, M^{\text{bg}}$ following E,

$$p(Y = 1|M^{\text{fg}}) \geq \tau \geq p(Y = 1|M^{\text{bg}}). \tag{7}$$

Therefore,

$$\begin{aligned} &P(p(Y = 1|M^{\text{fg}}) \geq \tau \mid T(M^{\text{fg}}) = 1) \\ &= P(p(Y = 1|M^{\text{bg}}) \leq \tau \mid T(M^{\text{bg}}) = 0) = 1 \end{aligned} \tag{8}$$

and so $\texttt{PxAcc}(p(Y|M), \tau) = 1$.

(Proof for "only if") Assume $\texttt{PxAcc}(p(Y|M), \tau) = 1$ for some τ. W.L.O.G., we assume that $P(T(M) = 1) \neq 0$ and $P(T(M) = 0) \neq 0$ (otherwise, $P(E) = 0$ and the statement is vacuously true). Then, Equation 8 must hold to ensure $\texttt{PxAcc}(p(Y|M), \tau) = 1$. Equation 7 then also holds almost surely, implying $\alpha \geq 1$ almost surely. ■

A.3. Foreground-background posterior ratio

We have described the the pathological scenario for WSOL as when the foreground-background posterior ratio α is small (§3.2). We discuss in greater detail what it means and whether there are data-centric approaches to resolve the issue. For quick understanding, assume the task is the localization of duck pixels in images. The foreground cue of interest M^{fg} is "feet" of a duck and background cue of interest M^{bg} is "water". Then, we can write the posterior ratio as

$$\alpha := \frac{p(\text{duck}|\text{feet})}{p(\text{duck}|\text{water})} = \frac{p(\text{feet}|\text{duck})}{p(\text{water}|\text{duck})} \cdot \left(\frac{p(\text{feet})}{p(\text{water})}\right)^{-1}$$

$\alpha < 1$ implies that lake patches are more abundant in duck images than are duck's feet (see Figure 1) for an illustration.

To increase α, two approaches can be taken. (1) Increase the likelihood ratio $\frac{p(\text{feet}|\text{duck})}{p(\text{water}|\text{duck})}$. This can be done by collecting more images where duck's feet have more pixels than lake does. (2) Decrease the prior ratio $\frac{p(\text{feet})}{p(\text{water})}$. Note that the prior ratio can be written

$$\frac{p(\text{feet})}{p(\text{water})} = \frac{p(\text{feet}|\text{duck})p(\text{duck}) + p(\text{feet}|\text{duck}^c)p(\text{duck}^c)}{p(\text{water}|\text{duck})p(\text{duck}) + p(\text{water}|\text{duck}^c)p(\text{duck}^c)}$$

Method	Paper	Code
CAM [13]	$\overline{s}_{ij} \geq 0.2$	$\overline{s}_{ij} \geq 0.2$
HaS [4]	Follow CAM†	Follow CAM
ACoL [11]	Follow CAM	$\widehat{s}_{ij} \geq$ unknown
SPG [12]	Grid search threshold	$\widehat{s}_{ij} \geq$ unknown
ADL [2]	Not discussed	$\widehat{s}_{ij} \geq 0.2^\dagger$
CutMix [9]	$\overline{s}_{ij} \geq 0.15$	$\widehat{s}_{ij} \geq 0.15$
Our protocol	$\widehat{s}_{ij} \geq \tau^\star$	$\widehat{s}_{ij} \geq \tau^\star$

$$\overline{s}_{ij} := \frac{s_{ij}}{\max_{kl} s_{kl}} \qquad \widehat{s}_{ij} := \frac{s_{ij} - \min_{kl} s_{kl}}{\max_{kl} s_{kl} - \min_{kl} s_{kl}}$$

Table 1: **Calibration and thresholding in WSOL.** Score calibration is done per image: max ($\overline{s}_{ij}$) or min-max ($\widehat{s}_{ij}$) normalization. Thresholding is required only for the box evaluation. $\tau^\star$ is the optimal threshold (§4.1 in main paper). Daggers (†) imply that the threshold depends on the backbone architecture.

With fixed likelihoods $p(\text{feet}|\text{duck})$ and $p(\text{water}|\text{duck})$, one can decrease the prior ratio by increasing the likelihood of lake cues in non-duck images $p(\text{water}|\text{duck}^c)$. We can alter WSOL into a more well-posed task also by including many background images containing confusing background cues.

Such data-centric approaches are promising future research directions for turning WSOL into a well-posed task.

B. Evaluation Protocol for WSOL

B.1. Score map normalization

A common practice in WSOL is to normalize the score maps per image because the maximal (and minimal) scores differ vastly across images. Prior WSOL papers have introduced either max normalization (dividing through by $\max_{ij} s_{ij}$) or min-max normalization (additionally mapping $\min_{ij} s_{ij}$ to zero). After normalization, WSOL methods threshold the score map at τ to generate a tight box around the binary mask $\{(i, j) \mid s_{ij} \geq \tau\}$. τ is typically treated as a fixed value [13, 11, 9] or a hyperparameter to be tuned [4, 12, 2]. We summarize that how prior works calibrate and threshold score maps in Table 1. As discussed in §4.1 of main paper, we use the min-max normalization.

B.2. Data preparation

We present the following data-wise contributions in this paper (contributions **bolded**):

- CUB: **New data** (5 images per class) with **bounding box annotations**.
- ImageNet: ImageNetV2 [5] with new **bounding box annotations**.
- OpenImages: **Processed** a split for the WSOL training, hyperparameter search, and evaluation with ground truth object masks.

B.2.1 ImageNet

The *test set* of ImageNet-1k dataset [6] is not available. Therefore, many researchers report the accuracies on the *validation set* for their final results [9]. Since this practice may let models overfit to the evaluation split over time, ImageNetV2 [5] has been proposed as the new test sets for ImageNet-1k trained models. ImageNetV2 includes three subsets according to the sampling strategies, `MatchedFrequency`, `Threshold0.7`, and `TopImages`, each with 10 000 images (10 images per class). We use the `Threshold0.7` split as our `train-fullsup`. Since ImageNetV2 does not contain localization supervision, we have annotated bounding boxes on those images, following the annotation protocol of ImageNet. The total number of annotated bounding boxes are 18 532.

B.2.2 CUB

We have collected 5 images for each of the 200 CUB fine-grained bird classes from Flickr. The overall procedure is summarized as:

1. Crawl images from Flickr.
2. De-duplicate images.
3. Manually prune irrelevant images (three people).
4. Prune with model classification scores.
5. Resize images.
6. Annotate bounding boxes.

Crawl images from Flickr. Since original CUB itself is collected from Flickr, we use Flickr as the source of bird images. We have used the `Flickr API`[1] to crawl up to 400 images per class with class name as search terms. We have only crawled images under the following licenses:

- `Attribution`
- `Attribution-NonCommercial`
- `Public Domain Dedication`
- `Public Domain Mark`

De-duplicate images. We de-duplicate the images using the `ImageHash` library[2] first among the crawled images themselves and then against the training and test splits of CUB.

[1] https://www.flickr.com/services/api/
[2] https://pypi.org/project/ImageHash/

Figure 2: **CUB version 2.** Sample images.

Manually prune irrelevant images (three people). Since crawled images of each class contain negative-class images (non birds and birds of wrong categories), three humans have participated in the prune-out process. We have only kept the images that all three have voted for "positive".

Prune with model classification scores. To match with the original CUB data distribution, we have trained a fine-grained bird classifier using ResNet50 [3] on the CUB training split. Among crawled images, those with ground truth class confidence scores lower than 0.5 have been pruned out.

Achieving 5 images per class. At this point, most classes have more than 5 images per class and a few have less than 5. In the former case, we have randomly sampled 5 images per class.

Resize images. Photographic technologies have made significant advances over the decade since the time original CUB was collected. As a result, image size statistics differ. To fix this, we have resized the images appropriately.

Annotate bounding boxes. The evaluation on CUB is based on bounding boxes. Thus, the held-out set images are supplied with tight bounding boxes around birds (one per image).

B.2.3 OpenImages

There are three significant differences between OpenImagesV5 [1] and CUB or ImageNet that make the OpenImages not suitable as a WSOL benchmark in its original form. (1) Images are multi-labeled; it is not sensible to train classifiers with the standard softmax cross-entropy loss assuming single label per image. (2) OpenImages has less balanced label distributions. (3) There are nice instance segmentation masks, but they have many missing instances.

We have therefore processed a subset of OpenImages into a WSOL-friendly dataset where the above three issues are resolved. The procedure is as follows:

1. Prune multi-labeled samples.
2. Exclude classes with not enough samples.

Figure 3: **OpenImages V5 sample.** Example of "group-of" box annotations. There is only one "Book" mask in the left bottom corner.

3. Randomly sample images for each class.
4. Prepare binary masks.
5. Introduce ignore regions.

Prune multi-labeled samples. We prune the multi-labeled samples in the segmentation subset of OpenImages. This process rejects 34.5% samples in OpenImages.

Exclude classes with not enough samples. After pruning multi-label samples, some classes are not available in validation and test sets. We have first defined the minimum number of samples per classes as (300, 25, 50) for (train, validation, test), and have excluded classes that do not meet the minimum requirement, resulting in 100 classes (out of 350 original classes).

Randomly sample images for each class. We have randomly sampled (300, 25, 50) samples per class for (train, validation, test). This eventually results in 29 819 `train-weaksup` (train), 2 500 `train-fullsup` (validation), and 5 000 `test` (test) samples.

Prepare binary masks. WSOL methods are evaluated against binary masks of the foreground category in each image. Since OpenImages comes with instance-level masks of various categories, we have taken the union of instance masks of the class of interest in every image to build the binary masks.

Introduce ignore regions. OpenImages has instance-wise masks, but not all of them are annotated as masks. When the objects are difficult to be annotated instance-wise or there are simply too many, the "group-of" box annotations are provided over the region containing multiple instances (*e.g.* hundreds of books on bookshelves in an image taken at a library – See Figure 3). We have indicated the regions with the "group-of" boxes as "ignore" regions, and have excluded them from the evaluation during the computation of `PxPrec`, `PxRec`, and `PxAP`.

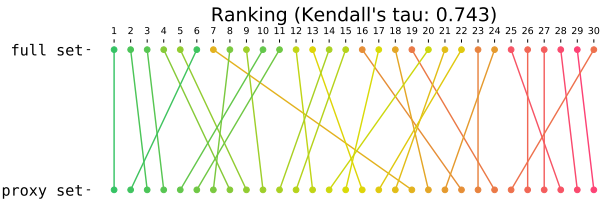


Figure 4: **Proxy ImageNet ranking**. Ranking of hyperparameters is largely preserved between the models trained on the full `train-weaksup` and its 10% proxy. Kendall's tau is 0.743.

B.3. Transferability of rankings between `train-fullsup` and `test`

As detailed in main paper §4.2, we search hyperparameters on `train-fullsup` and test them on `test`. To validate if the found hyperparameter rankings do transfer well between the splits, we show the preservation of ranking statistics in Table 2 and visualize the actual rankings in Figure 6. We observe that the rankings are relatively well-preserved (with Kendall's tau values > 0.7).

B.4. Hyperparameter search with proxy `train-weaksup`

We have reduced the training set size (`train-weaksup`) to 10% for ImageNet hyperparameter search for the interest of computational efficiency (§4.2). We examine how much this reduction affects the rankings of hyperparameters. We show a similar analysis as in §B.3: see Figure 4. We observe again that with Kendall's tau value 0.743, the two rankings are largely preserved.

C. Experiments

C.1. Prior WSOL methods and hyperparameters

We describe each method in greater detail here. The list of hyperparameters for each method is in Table 3.

CAM, CVPR'16 [13]. Class Activation Mapping trains a classifier of fully-convolutional backbone with global average pooling structure, and at test time uses the logit outputs before the global average pooling for the scoring rule $s(X_{ij})$. See Appendix §A.1 for the interpretation of CAM as a posterior approximator. CAM has learning rate (LR) and the score-map resolution (SR) as hyperparameters. LR is sampled log-uniformly from $[10^{-5}, 10^{0}]$, where end points correspond roughly to "no training" and "training always diverges" cases. SR is sampled from Categorical$\{14, 28\}$, two widely used resolutions in prior

Methods	ImageNet				CUB				OpenImages				Total
	VGG	Inception	ResNet	Mean	VGG	Inception	ResNet	Mean	VGG	Inception	ResNet	Mean	Mean
CAM [13]	0.887	0.795	0.933	0.872	0.731	0.706	0.758	0.732	0.869	0.714	0.934	0.839	0.814
HaS [4]	0.855	0.795	0.867	0.839	0.422	0.714	0.867	0.668	0.786	1.000	0.667	0.818	0.775
ACoL [11]	0.981	1.000	0.619	0.867	0.895	0.850	0.850	0.854	0.983	0.970	0.956	0.970	0.897
SPG [12]	0.944	0.766	0.900	0.870	0.697	0.779	0.889	0.788	0.758	0.874	0.929	0.854	0.837
ADL [2]	0.917	0.944	0.857	0.906	0.891	1.000	0.810	0.900	0.891	1.000	1.000	0.964	0.923
CutMix [9]	0.897	0.571	1.000	0.823	0.544	0.407	0.879	0.610	0.838	0.867	0.714	0.806	0.746

Table 2: **In-distribution ranking preservation.** Kendall's tau values for the hyperparameter ranking between `train-fullsup` and `test` are shown.

Methods	Hyperparameter	Distribution
Common	Learning rate	LogUniform$[10^{-5}, 10^{0}]$
	Score-map resolution	Categorical$\{14, 28\}$
HaS [4]	Drop rate	Uniform$[0, 1]$
	Drop area	Uniform$[0, 1]$
ACoL [11]	Erasing threshold	Uniform$[0, 1]$
SPG [12]	Threshold δ_l^{B1}	Uniform$[0, 1]$
	Threshold δ_h^{B1}	Uniform$[\delta_l^{B1}, 1]$
	Threshold δ_l^{B2}	Uniform$[0, 1]$
	Threshold δ_h^{B2}	Uniform$[\delta_l^{B2}, 1]$
	Threshold δ_l^{C}	Uniform$[0, 1]$
	Threshold δ_h^{C}	Uniform$[\delta_l^{C}, 1]$
ADL [2]	Drop rate	Uniform$[0, 1]$
	Erasing threshold	Uniform$[0, 1]$
CutMix [9]	Size prior	$\frac{1}{\text{Uniform}(0,2]} - \frac{1}{2}$
	Mix rate	Uniform$[0, 1]$

Table 3: **Hyperparameter search spaces.**

WSOL methods. All five methods below use CAM technique in the background, and have LR and HR as design choices.

HaS, ICCV'17 [4]. Hide-and-Seek (HaS) is a data augmentation technique that divides an input image into grid-like patches, and then randomly select patches to be dropped. The hyperparameters of HaS are drop rate (DR) and drop area (DA). Specifically, the size of each patch is decided by DA, and the probability of each patch to be selected for erasing is decided by DR. DA is sampled from a uniform distribution $U[0, 1]$, where 0 corresponds to "no grid" and 1 indicates "full image as one patch".

ACoL, CVPR'18 [11]. Adversarial Complementary Learning (ACoL) adds one more classification head to backbone networks. From one head, ACoL finds the high-score region using CAM using erasing threshold (ET) and erases it from an internal feature map. The other head learns remaining regions using the erased feature map. We sample ET from a uniform distribution $U[0, 1]$, where 0 means "erasing whole feature map" and 1 means "do not erase".

SPG, ECCV'18 [12]. Self-produced Guidance (SPG) utilizes spatial information about fore- and background using three additional branches (`SPG-B1,B2,C`). To divide foreground and background from score-map, they introduce two hyperparameters, δ_l and δ_h, per each branch. When the score is lower than δ_l, the pixel is considered as background, and the pixel is considered as foreground when the score is higher than δ_h. The remaining region (higher than δ_l, lower than δ_h) is ignored. We first sample δ_l from $U[0, 1]$, and then δ_h is sampled from $U[\delta_l, 1]$.

ADL, CVPR'19 [2]. Attention-based Dropout Layer (ADL) is a block applied on an internal feature map during training. ADL produces a drop mask by finding the high-score region to be dropped using another scoring rule [10]. Also, ADL produces an importance map by normalizing the score map and uses it to increase classification power of the backbone. At each iteration, only one component is applied between the drop mask and importance map. The hyperparameters of ADL are drop rate (DR) that indicates how frequently the drop mask is selected and erasing threshold (ET) that means how large regions are dropped. We sample DR and ET from uniform distributions $U[0, 1]$.

CutMix, ICCV'19 [9]. CutMix is a data augmentation technique, where patches in training images are cut and paste to other images and target labels are mixed likewise. Its hyperparameters consist of the size prior α (used for sampling sizes according to $\sim$ Beta(α, α)) and the mix rate r (Bernoulli decision for "CutMix or not"). The size prior is sampled from the positive range $\frac{1}{\text{Unif}(0,2]} - \frac{1}{2}$; then, Var(Beta$(\alpha, \alpha)$) follows the uniform distribution between 0 and 0.25 (maximal variance; two Dirac deltas at 0 and 1).

C.2. Classification results of WSOL methods

The widely-used "top-1 localization accuracy" for WSOL represents both the classification *and* localization performances. The metric can be misleading as a localization metric, as the increase in numbers can also be attributed to the improved classification accuracies. Thus, in the main paper, we have suggested using on the "GT-known" type

Methods	ImageNet				CUB				OpenImages				Total
	VGG	Inception	ResNet	Mean	VGG	Inception	ResNet	Mean	VGG	Inception	ResNet	Mean	Mean
CAM [13]	63.8	68.7	75.9	69.5	26.8	61.8	58.4	49.0	67.3	36.6	72.6	58.8	59.1
HaS [4]	61.9	65.5	63.1	63.5	70.9	69.9	74.5	71.8	60.0	68.4	74.0	67.5	67.6
ACoL [11]	60.3	64.6	61.6	62.2	56.1	71.6	64.0	63.9	68.2	40.7	70.7	59.9	62.0
SPG [12]	61.6	65.5	63.4	63.5	63.1	58.8	37.8	53.2	71.7	43.5	65.4	60.2	59.0
ADL [2]	60.8	61.6	64.1	62.2	31.1	45.5	32.7	36.4	66.1	46.6	56.1	56.3	51.6
CutMix [9]	62.2	65.5	63.9	63.8	29.2	70.2	55.9	51.8	68.1	53.1	73.7	65.0	60.2

Table 4: **Classification performance of WSOL methods.** Classification accuracies of the models in Table 2 of the main paper are shown. Hyperparameters for each model are optimally chosen for the localization performances on `train-fullsup` split. Classification performances may be sub-optimal when the best localization performance is achieved at early epochs.

		Top-1 localization accuracy						GT-known localization; `MaxBoxAcc` and `PxAP`								
		ImageNet			CUB			ImageNet			CUB			OpenImages		
	Methods	V	I	R	V	I	R	V	I	R	V	I	R	V	I	R
Reported	CAM [13]	42.8	-	46.3	37.1	43.7	49.4	-	62.7	-	-	-	-	-	-	-
	HaS [4]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	ACoL [11]	45.8	-	-	45.9	-	-	-	-	-	-	-	-	-	-	-
	SPG [12]	-	48.6	-	-	46.6	-	-	64.7	-	-	-	-	-	-	-
	ADL [2]	44.9	48.7	-	52.4	53.0	-	-	-	-	75.4	-	-	-	-	-
	CutMix [9]	43.5	-	47.3	-	52.5	54.8	-	-	-	-	-	-	-	-	-
Reproduced	CAM [13]	45.5	48.8	51.8	45.8	40.4	56.1	61.1	65.3	64.2	71.1	62.1	73.2	58.1	61.4	58.0
	HaS [4]	46.3	49.7	49.9	55.6	41.1	60.7	61.9	65.5	63.1	76.2	57.7	78.1	56.9	58.5	58.2
	ACoL [11]	45.5	49.9	47.4	44.8	46.8	57.8	60.3	64.6	61.6	72.3	59.5	72.7	54.6	63.0	57.8
	SPG [12]	44.6	48.6	48.5	42.9	44.9	51.5	61.6	65.5	63.4	63.7	62.7	71.4	55.9	62.4	57.7
	ADL [2]	44.4	45.0	51.5	39.2	35.2	41.1	60.8	61.6	64.1	75.6	63.3	73.5	58.3	62.1	54.3
	CutMix [9]	46.1	49.2	51.5	47.0	48.3	54.5	63.9	62.2	65.4	71.9	65.5	67.8	58.2	61.7	58.6

	Architecture
V	VGG-GAP [7]
I	InceptionV3 [8]
R	ResNet50 [3]

Table 5: **Previously reported vs our results.** The first six rows are reported results in prior WSOL papers. When there are different performance reports for the same method in different papers, we choose the greater performance. The last six rows are our re-implemented results under the new evaluation method (§4).

of metrics like `MaxBoxAcc` and `PxAP` that measures the localization performances given perfect classification.

Here, we measure the classification accuracies of the models in Table 2 of the main paper, to complete the analysis. The performances are reported in Table 4. There are in general great fluctuations in the classification results (26.8% to 74.5% on CUB). This is because we have selected the hyperparameters for each model that maximize the localization performances on `train-fullsup` split. In many cases, the best localization performances are achieved at early epochs, before the classifiers are sufficiently trained (see also §C.6 and Figure 12). The result signifies that localization and classification performances may not necessarily correlate and, therefore, localization-only metrics like `MaxBoxAcc` and `PxAP` must be used for model selection and evaluation of WSOL methods.

C.3. Reproducing prior WSOL results

We also summarize reported results in prior WSOL papers [13, 4, 11, 12, 2, 9] along with our reproduced results in Table 5. While our main evaluation metrics are the classification-disentangled GT-known measures (§4.1 in main paper), we also report the *top-1 localization* metrics that also measure the classification scores to match the reported numbers. Note that we use oracle τ for Top-1 localization accuracy.

We experimentally observe that training epochs and batch sizes influence the localization accuracies ($\sim$ 10pp in `MaxBoxAcc`). However, the training details for each method are not given by the corresponding papers. All methods should share the same training budget for fair comparisons. Therefore, we fix our training epochs to (10, 50, 10) for (ImageNet, CUB, OpenImages) experiments, and fix the batch size to 32 for all datasets.

Following the prior methods [13, 4, 11, 12, 2, 9], we train all six methods by fine-tuning ImageNet pre-trained model. Specifically, we apply $10\times$ scale of learning rate to higher-level layers of backbone networks during training. We consider the last two layers for VGG, last three layers for InceptionV3, and last three residual blocks, and fully connected

Methods	ImageNet (MaxBoxAccV2)				CUB (MaxBoxAccV2)				OpenImages (PxAP)				Total
	VGG	Inception	ResNet	Mean	VGG	Inception	ResNet	Mean	VGG	Inception	ResNet	Mean	Mean
CAM [13]	60.0	63.4	63.7	62.4	63.7	56.7	63.0	61.1	58.3	63.2	58.5	60.0	61.2
HaS [4]	+0.6	+0.3	-0.3	+0.2	+0.0	-3.3	+1.7	-0.5	-0.2	-5.1	-2.6	-2.6	-1.0
ACoL [11]	-2.6	+0.3	-1.4	-1.2	-6.3	-0.5	3.5	-1.1	-4.0	-6.0	-1.2	-3.7	-2.0
SPG [12]	-0.1	-0.1	-0.4	-0.2	-7.4	-0.8	-2.6	-3.6	+0.0	-0.9	-1.8	-0.9	-1.6
ADL [2]	-0.2	-2.0	+0.0	-0.7	+2.6	+2.1	-4.6	+0.0	+0.4	-6.4	-3.3	-3.1	-1.3
CutMix [9]	-0.6	+0.5	-0.4	-0.2	-1.4	+0.8	-0.2	-0.3	-0.2	-0.7	-0.8	-0.6	-0.3
Best WSOL	60.6	63.9	63.7	62.6	66.3	58.8	66.4	61.1	58.7	63.2	58.5	60.0	61.2
FSL baseline	60.3	65.3	66.3	64.0	71.6	86.6	82.4	80.2	65.9	74.1	74.4	71.5	71.9
Center baseline	52.5	52.5	52.5	52.5	59.7	59.7	59.7	59.7	45.8	45.8	45.8	45.8	52.3

Table 6: **Re-evaluating WSOL with MaxBoxAccV2**. We re-evaluate six recently proposed WSOL methods with MaxBoxAccV2. The experimental setting is the same as that of Table 2 in the main paper.

layers for ResNet as the higher-level layers.

Note that with GT-known metrics, our re-implementations produce better performances than the previously reported results (*e.g.* 62.7 $\rightarrow$ 65.3 for CAM on ImageNet with Inception backbone). This is perhaps because MaxBoxAcc and PxAP are based on the best operating thresholds.

Top-1 localization accuracies are reproduced well in general, except for the ADL: *e.g.* 52.4 $\rightarrow$ 39.2 for ADL on CUB with VGG backbone. This is due to the reduced training epochs compared to the original paper [2], which results in a decreased classification accuracies.

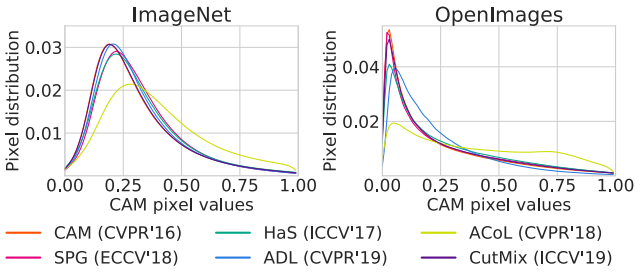


Figure 5: **CAM pixel value distributions.** On ImageNet and OpenImages test.

C.4. Score calibration and thresholding

The operating threshold τ for the score map s is an important parameter for WSOL. We show additional plots and visualizations to aid understanding. In Figure 7, we show the BoxAcc and PxPrec-PxRec performances at different operating thresholds with diverse architectures and datasets. It extends the main paper Figure 5. We observe again that the optimal operating thresholds $\tau^\star$ are vastly different across data and architectures for BoxAcc. The MaxBoxAcc in each method are relatively stable across methods especially on ImageNet. OpenImages PxPrec-PxRec curves do not exhibit big differences amongst WSOL methods, likewise.

In Figure 5, we visualize score distributions for different WSOL methods. We observe that methods have different distributions of scores; ACoL in particular tends to generate flatter score maps. Comparing dataset, we observe that OpenImages tends to have more peaky score distributions. It is therefore important to find the optimal operating point for each method and dataset for fair comparison.

We visualize more score maps in Figure 8, 9, 10; it extends the main paper Figure 4. We observe qualitatively different score maps in general across methods. However, the optimal IoU values are not as different and hard to predict just by visually looking at the samples. Again, it is important to have an objective way to set the thresholds τ in each case.

C.5. Hyperparameter analysis

We show the performance of all 30 hyperparameter search iterations on three datasets with three backbone architectures in Figure 11; it extends the main paper Figure 6. We observe similar trends as in the main paper. (1) Performances do vary according to the hyperparameter choice, so the hyperparameter optimization is necessary for the optimal performances. (2) CAM is among the more stable WSOL methods. (3) ACoL and ADL show greater sensitivity to hyperparameters in general. (4) CUB is a difficult benchmark where random choice of hyperparameters is highly likely to lead to performances worse than the center-Gaussian baseline.

C.6. Learning Curves.

We show the learning curves for CUB, ImageNet, OpenImages in Figure 12. We observe that the performances converge at epochs (10, 50, 10) for (ImageNet, CUB, OpenImages); the number of epochs for training models is thus sufficient for all methods. Note that learning rates are de-

cayed by 10 every (3, 15, 3) epochs for (ImageNet, CUB, OpenImages). For the most cases, performance increases as the training progresses. However, in some cases (*e.g.* CUB, SPG and OpenImages, CAM cases), best performances have already achieved at early iteration. That is, classification training rather hurts localization performance. Analyzing this is an interesting future study.

C.7. Evaluating WSOL methods with `MaxBoxAccV2`

We re-evaluate six recently proposed WSOL methods with `MaxBoxAccV2`, and the results are shown in Table 6. All training configurations but evaluation metrics are the same as those of the main paper. We evaluate the last checkpoint of each training session. We obtain the same conclusions as with the original metric, `MaxBoxAcc`: (1) there has been no significant progress in WSOL performances beyond vanilla CAM [13] and (2) with the same amount of fully-supervised samples, FSL baselines provide better performances than the existing WSOL methods.

References

[1] Rodrigo Benenson, Stefan Popov, and Vittorio Ferrari. Large-scale interactive object segmentation with human annotators. In *CVPR*, pages 11700–11709, 2019. 3

[2] Junsuk Choe and Hyunjung Shim. Attention-based dropout layer for weakly supervised object localization. In *CVPR*, pages 2219–2228, 2019. 2, 5, 6, 7

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3, 6

[4] Krishna Kumar Singh and Yong Jae Lee. Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization. In *ICCV*, pages 3524–3533, 2017. 2, 5, 6, 7

[5] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *ICML*, pages 5389–5400, 2019. 2, 3

[6] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 3

[7] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 6

[8] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, pages 2818–2826, 2016. 6

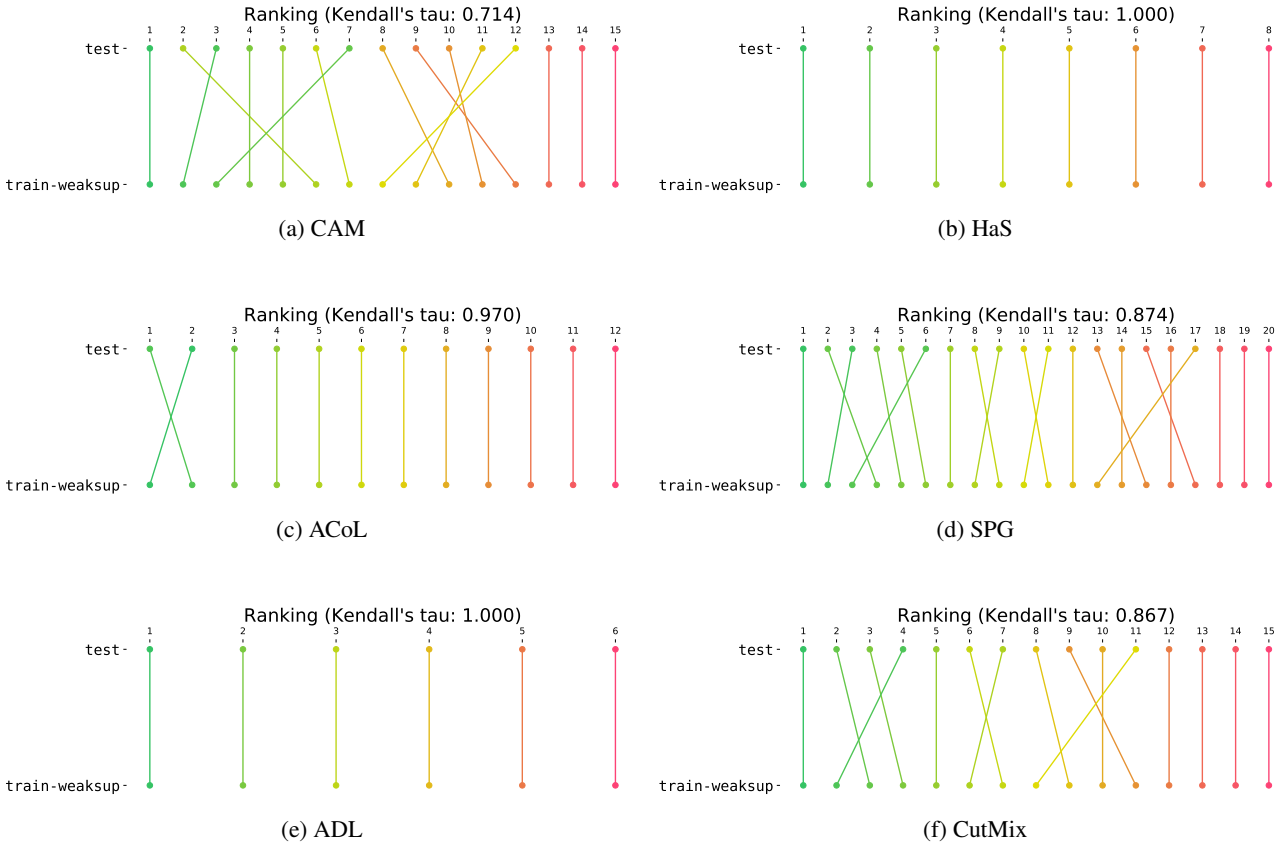
[9] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, 2019. 2, 3, 5, 6, 7

[10] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *ICLR*, 2017. 5

[11] Xiaolin Zhang, Yunchao Wei, Jiashi Feng, Yi Yang, and Thomas S Huang. Adversarial complementary learning for weakly supervised object localization. In *CVPR*, pages 1325–1334, 2018. 2, 5, 6, 7

[12] Xiaolin Zhang, Yunchao Wei, Guoliang Kang, Yi Yang, and Thomas Huang. Self-produced guidance for weakly-supervised object localization. In *ECCV*, pages 597–613, 2018. 2, 5, 6, 7

[13] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *CVPR*, pages 2921–2929, 2016. 1, 2, 4, 5, 6, 7, 8



(a) CAM (b) HaS

(c) ACoL (d) SPG

(e) ADL (f) CutMix

Figure 6: **Preservation of hyperparameter rankings.** Ranking of hyperparameters is largely preserved between `train-weaksup` and `test`. We only show the converged sessions (§5.4). Results on ResNet50, OpenImages.

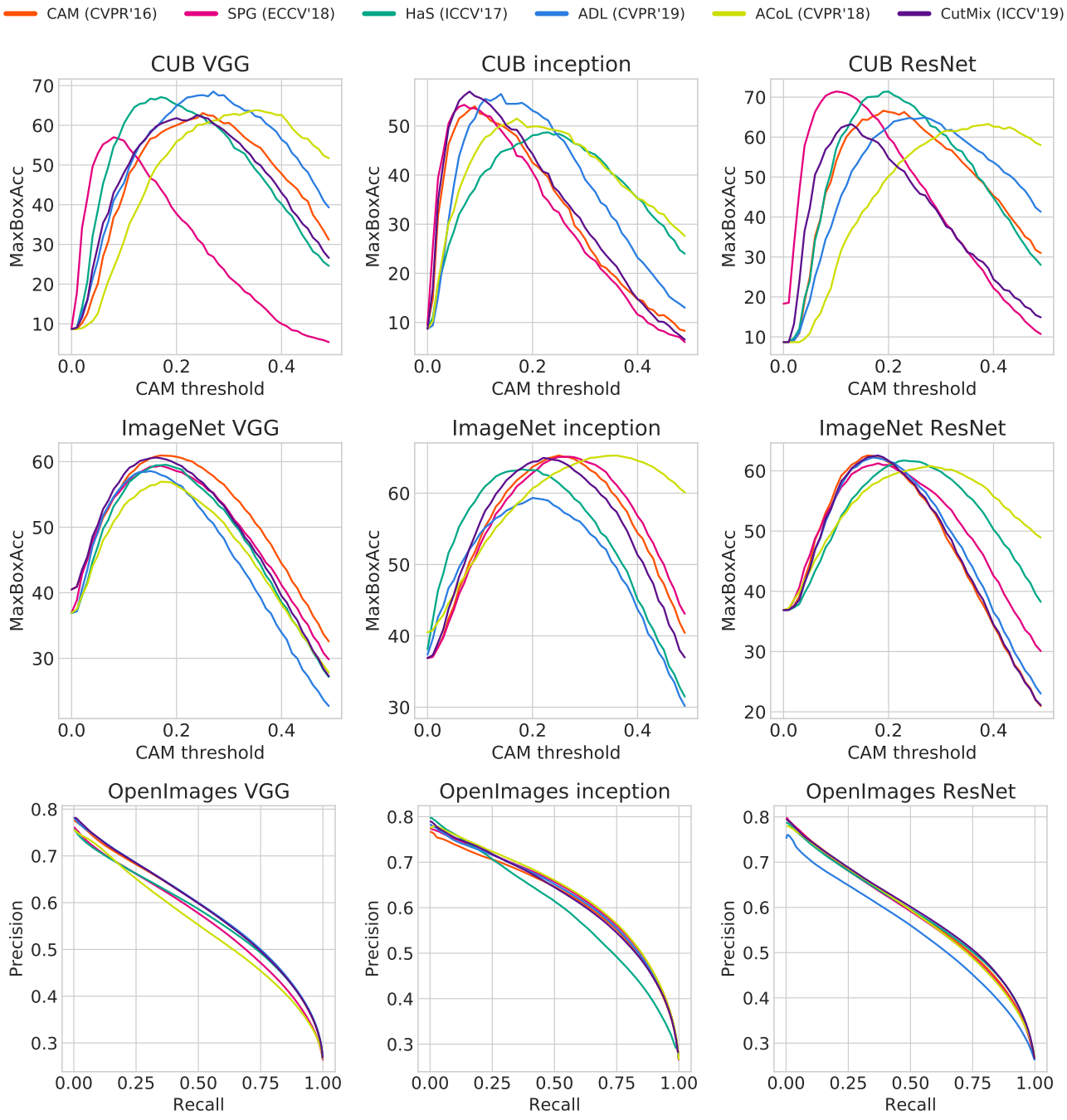
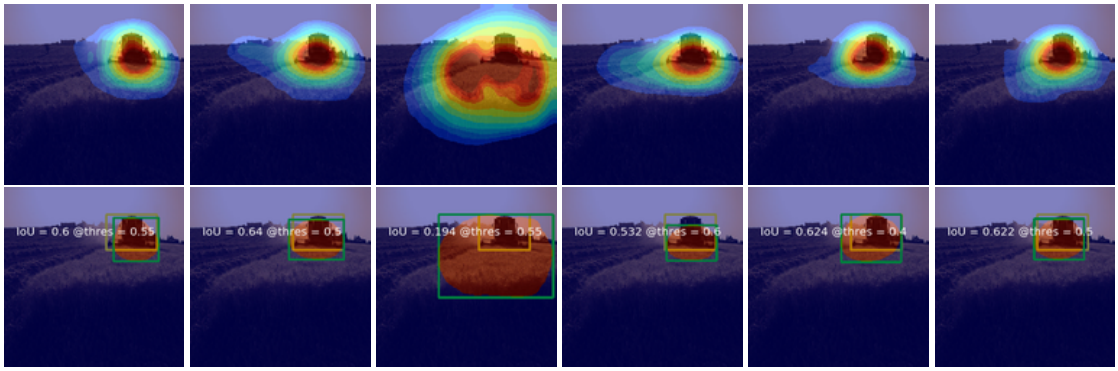
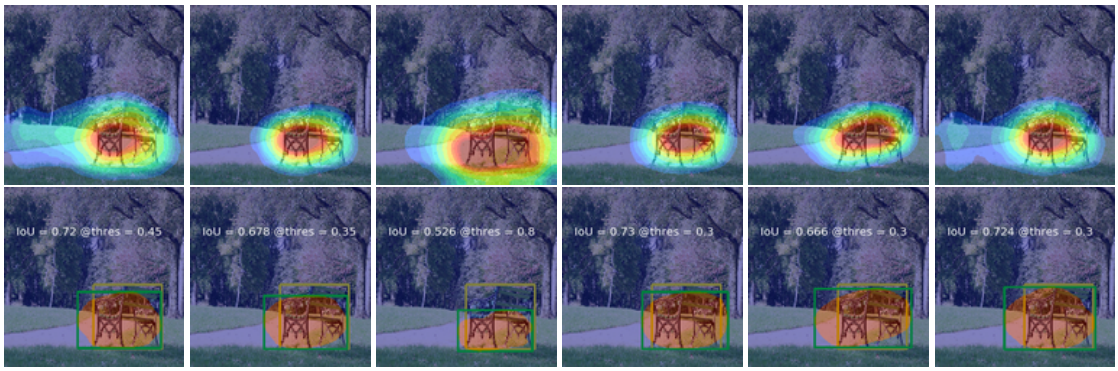


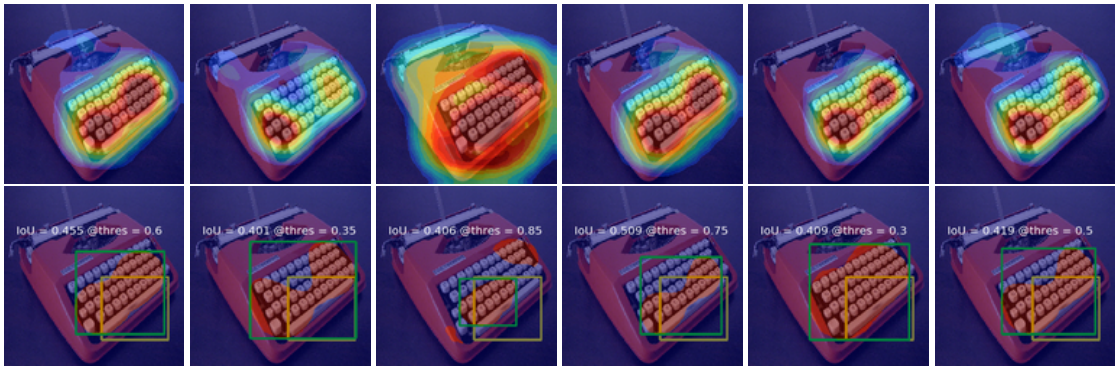
Figure 7: **Performance by operating threshold τ.** CUB and ImageNet: `BoxAcc` versus τ, OpenImages: `PxPrec` versus `PxRec`. ResNet, VGG, and Inception architecture results are used. This is the extension of Figure 5 in the main paper.



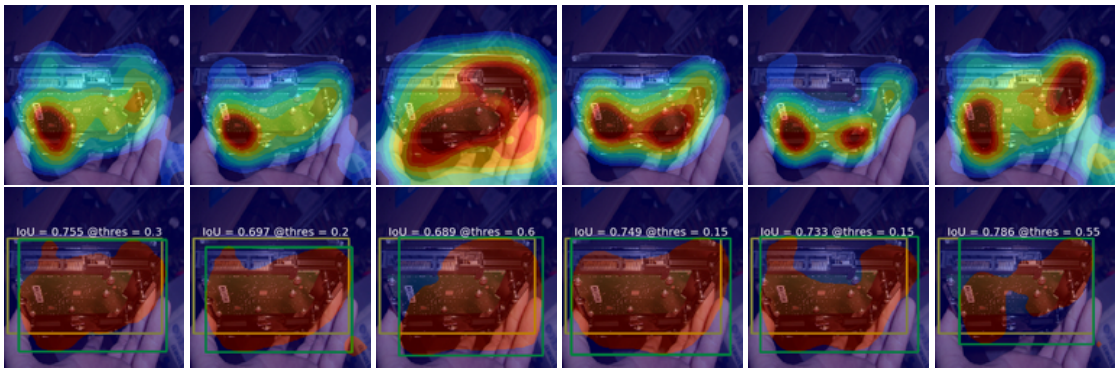
(a) Label: Harvester



(b) Label: Park bench

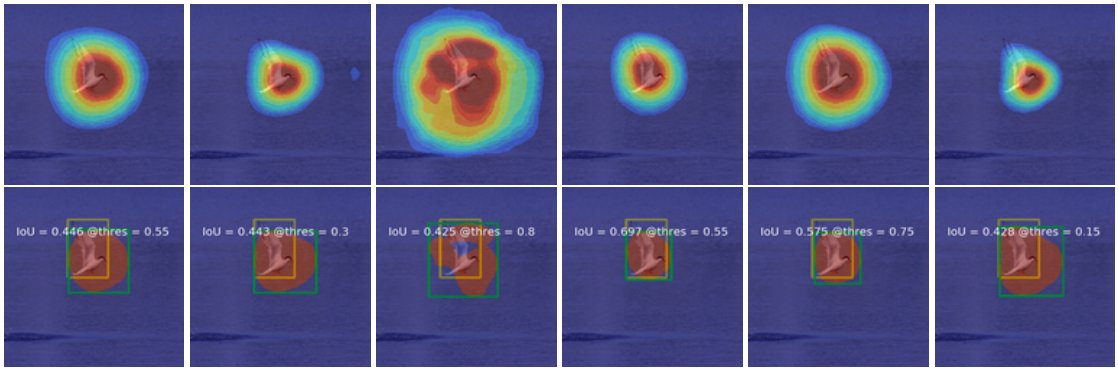


(c) Label: Space bar

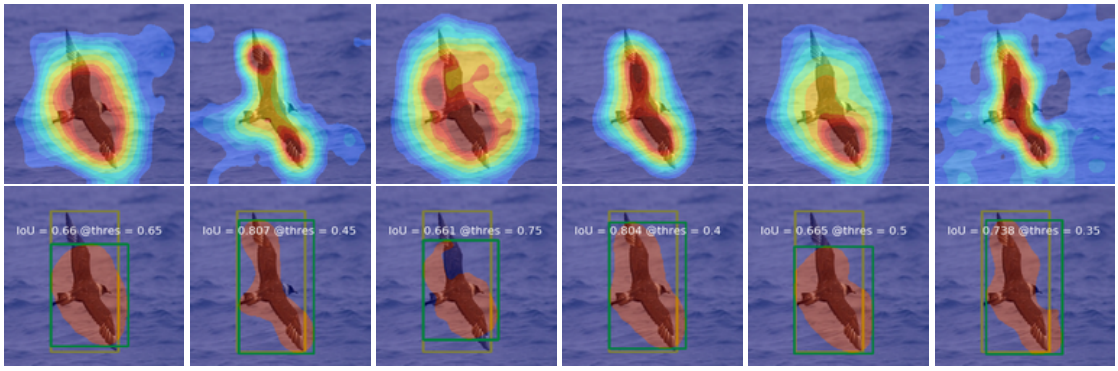


(d) Label: Hard disk

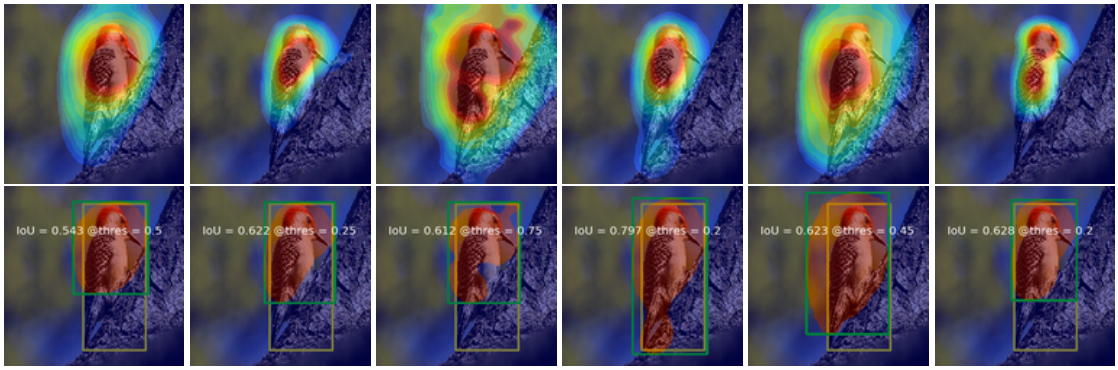
Figure 8: **ImageNet score maps.** Score maps of CAM, HaS, ACoL, SPG, ADL, CutMix from ImageNet.



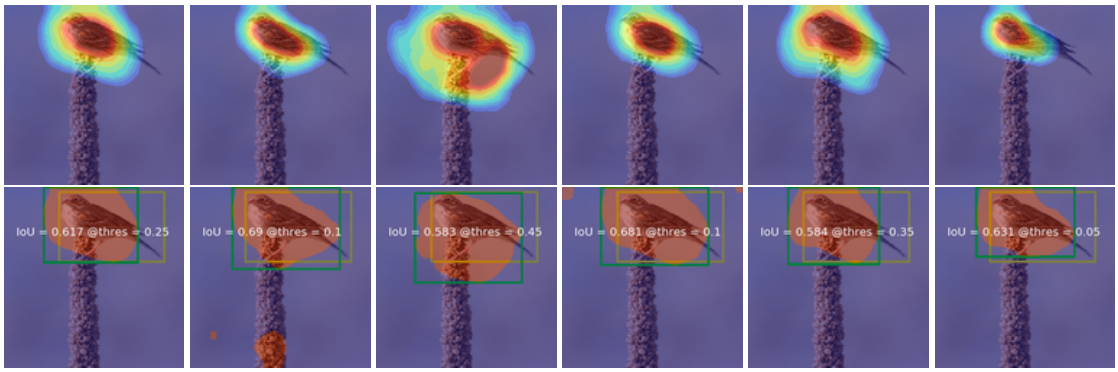
(a) Label: Common Tern



(b) Label: Pomarine Jaeger

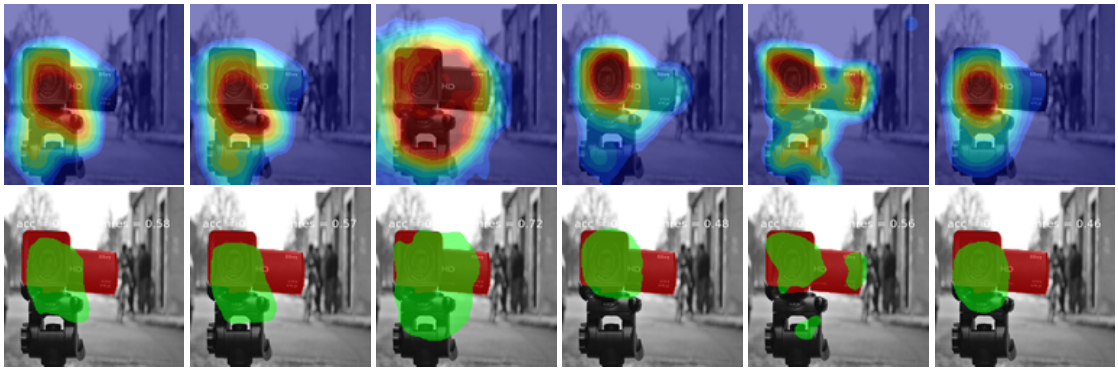


(c) Label: Red bellied Woodpecker

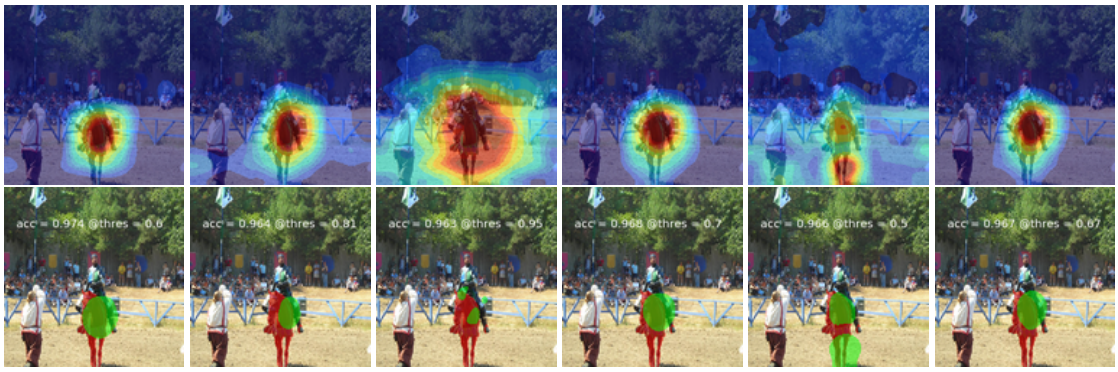


(d) Label: Vesper Sparrow

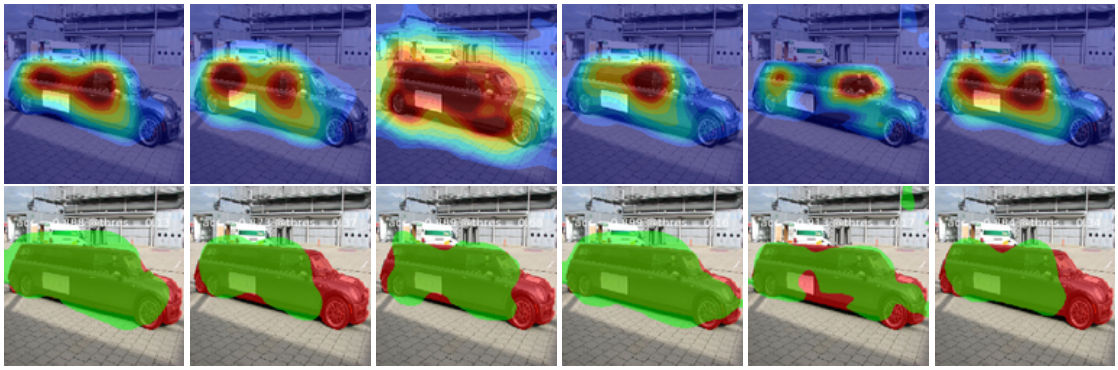
Figure 9: **CUB score maps.** Score maps of CAM, HaS, ACoL, SPG, ADL, CutMix from CUB.



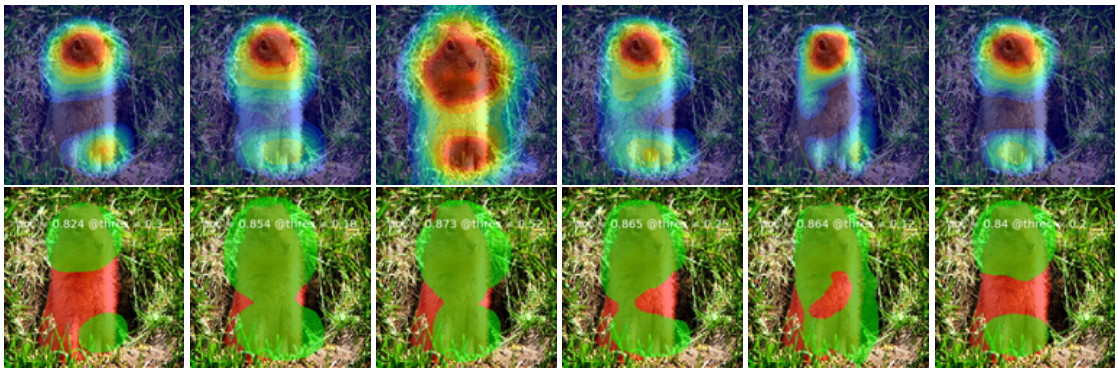
(a) Label: Camera



(b) Label: Horse

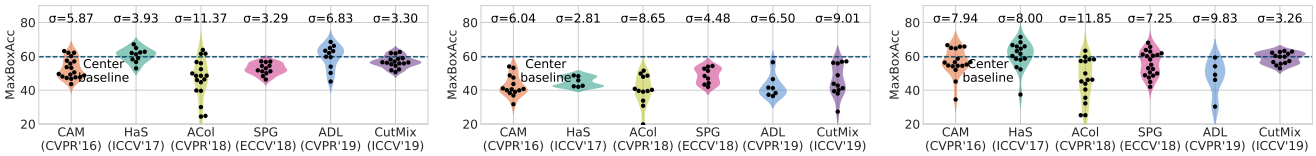


(c) Label: Limousine

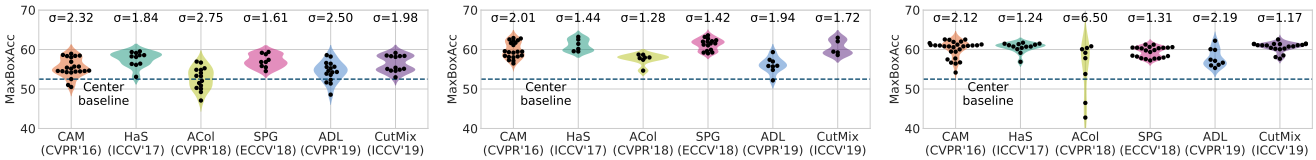


(d) Label: Squirrel

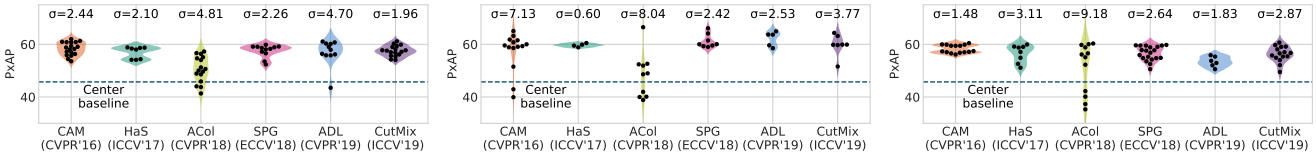
Figure 10: **OpenImages score maps.** Score maps of CAM, HaS, ACoL, SPG, ADL, CutMix from OpenImages.



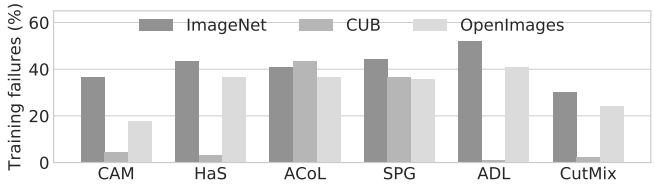
(a) CUB, VGG (b) CUB, Inception (c) CUB, ResNet



(d) ImageNet, VGG (e) ImageNet, Inception (f) ImageNet, ResNet



(g) OpenImages, VGG (h) OpenImages, Inception (i) OpenImages, ResNet



(j) Ratio of training failures

Figure 11: **All results of the 30 hyperparameter trials.** CUB, ImageNet, OpenImages performances of all 30 randomly chosen hyperparameter combinations for each method. This is the extension of Figure 6 in the main content.

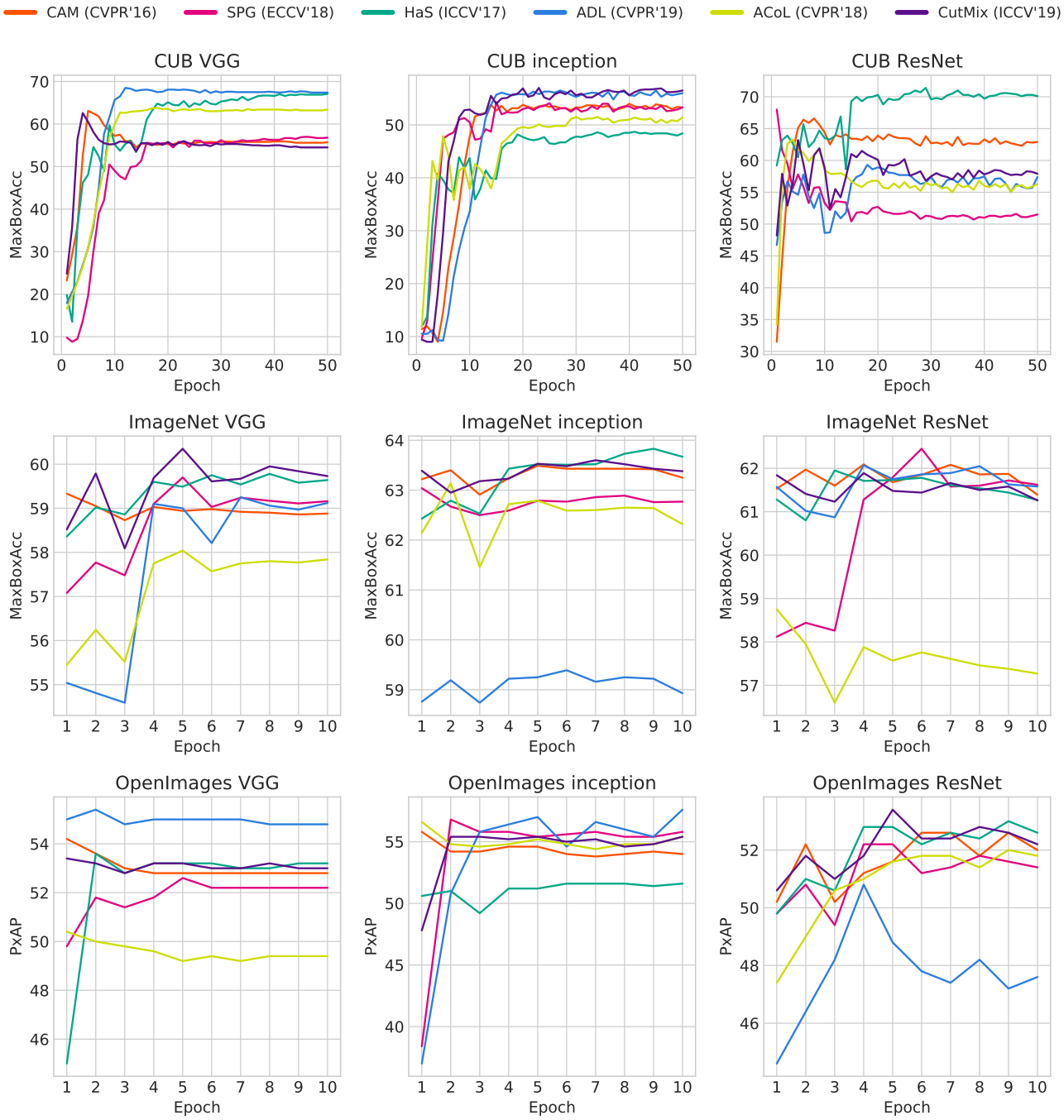


Figure 12: **Learning curves.** Results on three datasets with VGG, Inception and ResNet architectures.